Cloud-POA: A Cloud-Based Map Only Implementation of PO-MSA on Amazon Multi-node EC2 Hadoop Cluster

Nafis Neehal, Dewan Ziaul Karim, Ashraful Islam
Department of CSE
Daffodil International University
Dhaka, Bangladesh
{nafis.cse, ziaul.cse, ashraful.cse}@diu.edu.bd

Nafis Neehal, Dewan Ziaul Karim, Ashraful Islam
Department of CSE
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
{nafis.cse, ziaul.cse, ashraful.cse}@diu.edu.bd

***Abstract*—Sequence alignment in bioinformatics and computational biology has always been a challenging task. With Next Generation Sequencing (NGS) techniques in hand, researchers are now capable of studying biological systems at a level never been possible before. Scientists now have billions of bytes of biological data to work with, trillions of sequences to align. But this comes at a cost of requiring computing machines having a tremendous amount of computational and analytical power. Purchasing this huge amount of hardware and setting up a standalone infrastructure would not only cost an unnecessarily massive amount of money and labor but also would become troublesome to maintain. Moreover, for aligning a huge number of DNA or Protein sequences a scalable multiple sequence alignment (MSA) algorithms is needed with decent accuracy. In such context, this paper presents a novel implementation of Partial Order Alignment (POA) algorithm on a multi-node Hadoop Cluster running on MapReduce framework. The implementation was done in Amazon AWS platform with multiple EC2 instances. It is a map-only implementation with Hadoop Streaming. The result of this implementation shows a drastic reduction in runtime with no accuracy degradation.**

***Index Terms*—NGS, POA, PO-MSA, Hadoop, Mapreduce, Hadoop Streaming**

I. Introduction

Multiple Sequence Alignment (MSA) is a general extension of Pairwise alignment. When more than two biological sequences get aligned to each other, it is called multiple sequence alignment. The alignment can be based on DNA or Protein. There can be an evolutionary relationship among different query sequences in an input set. From this MSA, phylogenetic analysis can be done. Moreover, sequence homology can also be inferred.

The approach which is mostly used in terms of MSA uses a technique called progressive alignment. This is basically a heuristic search which is also known as the hierarchical or tree method. It was developed by Paulien Hogeweg Ben Hesper in 1984 [1]. Progressive alignments are not always globally optimal because of the errors made at any stage in growing MSA (does not matter whether its the beginning stage or any stage at the middle or end) and all those errors are carried into the final result. So this creates a problem in the accuracy of the result. Moreover, if all the sequences of the input set are distantly related, a significant decrease in the performance may occur. However, modern progressive methods usually modify their scoring function [2].

There are many progressive alignment methods. The Clustal family [3], especially the weighted variant ClustalW [4] is vastly popular. It is used for the construction of the phylogenetic tree. Another one which was in use is ClustalW2 (expired in August 2015). Later Clustal Omega was suggested which actually performs based on seeded guide trees and Hidden Markov Model (HMM) techniques for protein alignments. Various MSA tools for progressive DNA alignments are offered by them. Multiple Alignment using Fast Fourier Transform (MAFFT) [5] is one of them. Kalign is another decent quality MSA algorithm and it is similar to progressive alignment methods for sequence alignments.

There is another method which has a similar working procedure to progressive method but realigns the initial sequences as well as adding new sequences to the MSA constantly. This method is known as the iterative method. This procedure can decrease the errors inherent in progressive methods. Unlike progressive methods, iterative methods can go back to the pairwise alignments which were calculated earlier. It helps to optimize a general objective function, for example, finding a score which symbolizes high-quality alignment.

In various software packages, different types of iterative alignment algorithms have been implemented. Choosing a best technique is always tough although many comparisons and reviews have been done [6]. There are many software packages. PRRN/PRRP is one of them which uses hill climbing algorithm which helps in optimizing the MSA alignment score [7]. DIALIGN [8] is another iterative algorithm which focuses barely on local alignments without inaugurating a gap penalty. MUSCLE [9] is another popular iterative alignment algorithm. MUSCLE, which stands for Multiple Sequence Comparison by Log Expectation uses double distance measures Kmer Kimura. For distance measurement of unaligned pairs of sequences, Kmer is used. Whereas, Kimura is used for the aligned pair of sequences. Guide tree is built using UPGMA

978-1-5386-1150-0/17/$31.00 ©2017 IEEE

method.

For large scale sequence data sets, it is necessary for existing MSA algorithms to be executed in a parallel manner with sequence data to be distributed over various computing cluster.

In case of large sequence data sets the sequence data is needed to be distributed over several computing clusters and MSA algorithm is needed to be executed in a parallel manner. Thats where cloud computing plays its vital role. Though cloud computing is sometimes considered to provide only rental of computing power and storage, it actually provides different types of services too. The principle service models of cloud computing are Software as a service (SaaS), Infrastructure as a service (Iaas) and Platform as a service (Paas).

This paper presents a novel implementation of a very popular MSA algorithm namely Partial Order Alignment in the cloud platform. The implementation was done in Amazon AWS platform with multiple EC2 instances. A Hadoop Cluster was built and MapReduce framework was used in this implementation and the alignment job was run as a map only job. The result of this implementation shows a drastic reduction in runtime with no accuracy degradation.

The remaining part of the paper is arranged in the following manner - Related works done prior to this implementation is discussed in Section II. Details of POA algorithm is discussed in Section III. Hadoop Streaming and MapReduce Framework is Discussed in Section IV. Cloud-POA implementation details are described in Section V. Performance analysis is done in Section VI. Finally, conclusion with some future work scopes is described in Section VII and Section VIII.

II. Related Works

Implementing sequence alignment algorithms in cloud platform is a very recent trend. Lee et al [10] described a graphical representation of an MSA which can be aligned straightly by pairwise dynamic programming. Usually, progressive alignment methods tend to reduce an MSA to a linear profile for each of the alignment steps. But this results in some loss of information which eventually has a toll in accurate alignment. Lee et al banished the necessity to diminish the MSA to a profile. This also enabled POA to guarantee that the optimal alignment of each new sequence against each sequence in the MSA would be taken into account. This algorithm improved in comparison with other algorithms in terms of linear time complexity (e.g. an alignment of 5000 sequences in 4h on Pentium II), which enabled the construction of massive and complex alignments. The utility of this algorithm was demonstrated on a family of multi-domain SH2 proteins and on EST assemblies which contained alternative splicing and polymorphism.

Christopher Lee also described generating consensus sequence from POA in a paper published in 2003 [11]. It stated a dynamic programming algorithm name heaviest bundle for generating multiple consensus sequences. The degree of structural complexity of the source alignment is revealed by the number and relationships of these consensus sequences. The paper illustrated its value for inspecting expressed sequence alignments to identify alternative splicing, rebuild full-length mRNA isoform sequences from EST fragments. It also helped to distinguish paralog mixtures which can lead to inaccurate SNP predictions.

Ramu Chenna et al. [12] described the Clustal series of programs that are used for MSA of not only nucleic acid but also protein sequences. Eventually, it helps to build phylogenetic trees.

A review article [13] stated about the integration of different MSA methods with cloud computing. Next generation sequencing technologies are also changing the scenery of molecular biology. It results in huge amounts of raw sequence data which may eventually flood the databases. Since this ever-increasing sequence data sets create a noteworthy bottleneck, it is a burning necessity nowadays to implement the existing MSA algorithms in such a way so that they can run a parallel manner with sequence data dispersed over a computing cluster. When MSA algorithms get integrated with cloud computing technologies, speed is possibly going to improve. Same can be said in terms of quality and capability for MSA to work with a huge number of sequences.

Yu-Jung Chang et al. [14] described a next gen genomic sequence assembler based on MapReduce framework. The paper states CloudBrush a newly distributed genome assembler based on string graphs along with MapReduce framework. This assembler was evaluated against GAGE benchmarks to measure the assembly quality with other assemblers. The result showed that this new assembler had a moderately low (The assembler had a moderate N50) misassembly rate of misjoins and indels of >5 bp.

Michael C. Schatz et al. [15] created CloudBurst which uses the open source Hadoop implementation of MapReduce framework. It helps parallelizing the execution using multiple processing nodes. CloudBursts running time changes in a similar manner with the number of reads mapped. In a configuration with 24-processors, CloudBurst is up to 30 times faster than RMAP (executing on a single core).

Using cloud technologies in genomics and prepare for 2nd or 3rd generation DNA sequencing has always been a challenging task. For this purpose, a Hadoop MapReduce-based application named CloudAligner [16] was created which can achieve better performance, better accuracy and can provide a suitable user-friendly interface. CloudAligner omitted the reduce phase, thus a huge performance gain was observed over cloud-based counterparts (35 to 80 percent). It was seen from the experiment that CloudAligner easily outperformed CloudBurst from 35 to 67 percent.

The reason for choosing POA for implementation are some facts which should be taken into consideration. For example, Probcons, T-Coffee, Probalign, MAFFT were more accurate but consumed a huge amount of time and memory. On the other hand, ClustalW, Dialign, and MUSCLE were faster and less memory consuming, but less accurate. In case of POA, accuracy is quite good, and the runtime is moderate. So this is a kind of balanced algorithm compared to all the others. Since its accuracy is quite good and generates a moderate runtime, so

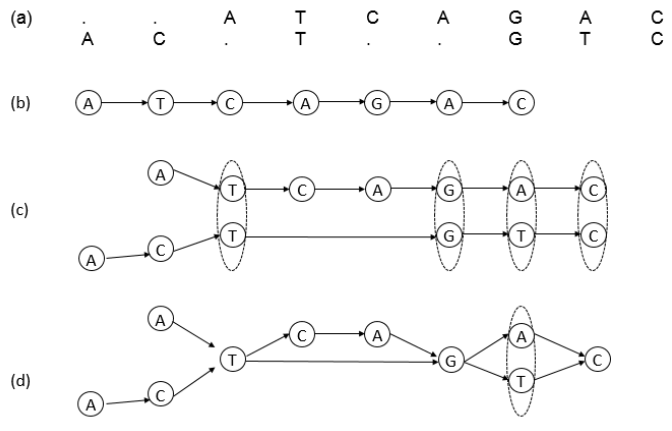

Fig. 1. POA Structure

if somehow the runtime can be decreased, then it can provide a really good outcome. There is a huge potential of decreasing the runtime radically through parallel implementation, which can eventually lead to better results than all other algorithms including CloudAligner or CloudBurst. Moreover, POA has a scalable structure in order to deploy for aligning large number sequences. All these considerations were taken into account while choosing POA for the cloud implementation.

III. Partial Order Alignment

POA is a graph based representation of an MSA algorithm [10] where nodes represent sequence letters and directed edges are drawn between two consecutive letters in each sequence. A series of nodes linked by directed edges can uniquely represent a single sequence.

If two sequences are given, PO-MSA structure can be drawn by redrawing the aligned version of these two sequences in PO-MSA format (Figure 1c). And if two identical letters are aligned, they are fused together as a single node. If two nonidentical letters are aligned, then they are represented as separate nodes but are remembered as being aligned to each other (Figure 1d). All the relevant information about previous nodes (ID of original sequences and index position of that letter in the original sequence) are stored in the newly generated node in case of node fusion. Hence from the PO-MSA, it is possible to reconstruct the original MSA and vice versa which leads to zero information loss and no degeneracy.

PO-MSA graph obeys linear ordering in regions where nodes have single outgoing edges. In case of linear ordering, for two distinct nodes i,j the ordering can be i $<$j XOR j $<$i where the ordering relation i $<$j represents that one or more paths of directed edges might exist between node i and j. But in case of partial ordering, such scenarios might occur where there is no directed edge from j to i OR i to j. Nodes with multiple incoming or outgoing edges are called junction nodes. A junction node has multiple branches defined

by different edges $edge_1$, $edge_2$, $edge_3$..... and so on. Two different nodes on different branches might not have any ordered relation to one another. So, PO-MSA data structure is basically a Directed Acyclic Graph (DAG).

POA is a simple extension of the Needleman-Wunsch algorithm which aligns a PO-MSA with a linear sequence. In case of POA, instead of using two linear sequences in two axes of the 2D matrix, one of the linear sequences is replaced by a partial order containing branching. Then the partial order sequence is transferred to the 2D matrix. On any given surface, POA shows similar behavior as a standard 2D alignment algorithm with the same three moves (diagonal, vertical and horizontal). Diagonal and Horizontal moves are somewhat extended to allow any of the incoming surfaces meeting at any junction node.

At a given cell (n,m) in the matrix, the scores for all possible moves are calculated and the move with the maximum score is selected and saved at this cell which is given by the following equation -

$$S(n,m) = max \begin{cases} S(p,m-1) + s(n,m) \\ S(p,m) + \Delta(m) \\ S(n,m-1) + \Delta(n) \end{cases} \quad (1)$$

while aligning residue n to residue m and the horizontal or vertical gap penalty is Δ considering all predecessor nodes p that has a directed edge from p $\rightarrow n$.

PO-MSA nodes are fused according to the (i_S, j_G) mapping as follows -

- If node i_S and j_G are aligned and have identical letters, then they are fused
- Else if node i_S and j_G are aligned but do not have identical letters and if j_G is already aligned to another node k_G whose letter is identical to i_S then i_S and k_G are fused
- Else i_S and j_G are recored to aligned with each other as separate nodes.

After this, finally, the redundant directed edges are removed and the final PO-MSA graph is constructed.

IV. Hadoop's MapReduce and Streaming

A. Apache Hadoop

Apache Hadoop is an open-source software framework. Distributed storage and processing of huge data sets for big data using MapReduce framework is done with hadoop. It consists of computer clusters built with different computer hardware rented by the authority to the clients. Hadoop has a storage system which is known as Hadoop Distributed File System (HDFS), and a processing system which is a MapReduce programming model. Hadoop splits single large files into chunks of smaller data blocks and distributes them across the nodes in a cluster which allows the data to be processed efficiently and faster.

B. Mapreduce Framework

MapReduce is a programming model for processing large data sets [17]. Users define a map function which processes a huge amount of data and generates key/value pair $<K_i,V_i>$as an intermediate data. A reduce function is also defined that merges all the intermediate values associated with the same intermediate key and produces the final output. MapReduce framework architecture is shown in Figure 2.

C. Hadoop Streaming

Hadoop streaming is a unique utility on top of Hadoop distribution. Programmers are now capable of creating and running MapReduce jobs written and exported as any type of executable or script as the mapper and the reducer. As Cloud-POA was implemented in Python 3.6, so Hadoop Streaming had to be used for running Python code on top of Hadoop.

V. Cloud-POA Implementation

Cloud-POA is a cloud based implementation of POA algorithm. It has been implemented on Amazon AWS platform. Ten Amazon EC2 nodes were used to build up a Multi-node Hadoop Cluster upon which the Cloud-POA framework has been built. The system architecture is shown in Figure 3.

The implementation is described in details in the following subsections -

A. Input Preprocessing

In this stage, the input file (FASTA) containing n DNA sequences was preprocessed by separating labels and sequences and storing them into two seperate lists namely `LABEL_LIST` = $label_1$, $label_2$, $label_3$.....,$label_n$ and `SEQUENCE_LIST` = seq_1, seq_2, seq_3.....,seq_n. Up to maximum 100k sequences were taken as input. Number of base pairs was between 150-200. Synthetic data was used for testing purpose in initial stage of implementation. Later on, benchmark data was used for validation with higher number of sequences.

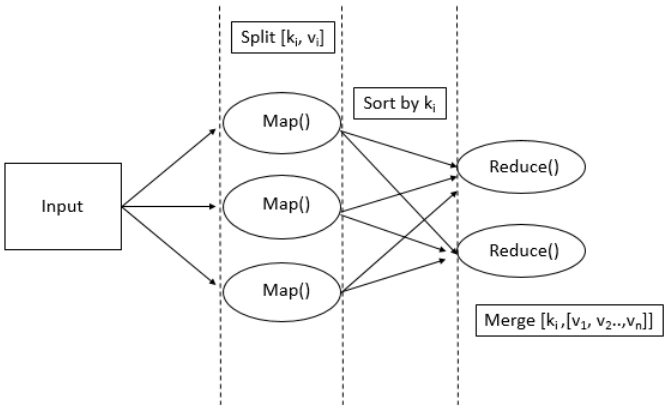

Fig. 2. MapReduce Framework Architecture

TABLE I
Cloud-POA Hadoop Configuration

Instance Name	Instance Type	Number of Machines	RAM, HDD, OS
Namenode	Master	1	4GB, 16GB, Ubuntu 16.04
Secondary Namenode	Master	1	4GB, 16GB, Ubuntu 16.04
Data Node	Slave	8	2GB, 4GB, Ubuntu 16.04

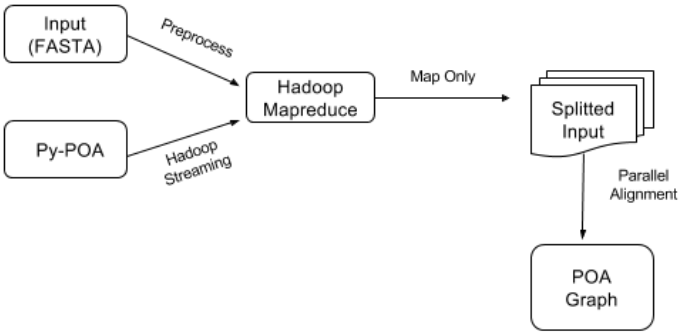

Fig. 3. Cloud POA Architecture

B. Hadoop Framework Configuration

Cloud-POA was implemented on Amazon AWS platform. For Cloud-POA, a Hadoop cluster made of 10 Amazon EC2 instances was built and MapReduce was installed on top of that. Framework details are given in Table 1.

C. Alignment and POA Graph Generation

After input has been preprocessed, POA algorithm was run over the input sequences. As Hadoop's MapReduce framework has been used for this implementation, the splitting of input sequences was done automatically by Hadoop's MapReduce. Default input split size (*mapred.min.split.size*) was set to 8 MB, so input files larger than 8 MB was automatically split and assigned to different mappers for performing alignment job. The POA Graph was declared as a global object, and from each split, each input was aligned parallelly to this POA Base Graph and POA Graph was updated. Algorithm 1 contains the pseudocode for the mapper implementation of Cloud-POA. As Cloud-POA is a map only implementation, there is no further processing and the output of mapper is the final output. In Mapper, there are 3 modules namely ReadFasta, POAGraph, and SeqGraphAlignment. ReadFasta does the preprocessing of input, POAGraph and SeqGraphAlignment do the POA graph generation part.

D. Consensus Sequences Generation

Finally, when all the alignments and insertions are finished and the final POA Graph is generated then multiple consensus sequences is generated by following the `HEAVIEST_BUNDLE` algorithm [11]. The pseudocode in Algorithm 2 shows the overall flow of the algorithm where G is the PO-MSA graph. In addition to generating consensus sequences C_1, C_2 . . .,C_n the top-level function `GENERATE_CONSENSI()` returns

their alignment (in G), and assigns each sequence S_k to a specific consensus (or to none at all). It adds a new sequence C_b to the PO-MSA to represent each consensus, which follows the maximum likelihood path P through G.

VI. Performance Analysis

Parallel implementation of POA algorithm in Amazon Elastic Compute Cloud (EC2) platform shows exemplary results. Figure 4 shows a drastic reduction of runtime in Cloud implementation of POA, unlike the standalone implementation. It roughly estimates that the runtime gets reduced around 65-70 percent.

Figure 5 shows a comparison between all major MSA algorithms in terms of runtime and here also Cloud-POA outperforms all other major MSA.

This comparison was done on GENIE gene finding Benchmark data set, containing a total of 793 human genes [19]

Figure 6 shows a portion of the final aligned sequences along with the consensus sequences which was done on the GENIE data set mentioned earlier.

VII. Future Work

In this paper, the POA algorithm which is used is sensitive to sequence alignment order. This can be a vital issue in some cases. One possible workaround for this issue is to use CLUSTAL-like progressive alignment algorithm. In other words, using a guide tree to establish the best order for aligning the sequences. This procedure requires one set of aligned sequences into another. So this needs expanding the POA algorithm to align two PO-MSAs to each other. Moreover, a good guide tree can improve estimated phylogenies [18].

The algorithm used in this paper usually aligns the base POA graph with the new DNA sequence graph at the very first. Then the resultant graph is aligned with another new DNA sequence graph. This procedure continues in an iterative way. But there is another way to execute this. Consider taking two sequences and build graphs from them. Now align these two graphs and we find a resultant graph. Consider naming it as G_1 (Graph1). Now for the next two sequences, do the same and consider naming the new resultant graph as G_2 (Graph2). G_3, G_4.....,G_n will follow the same procedure. Then if somehow G_1 and G_2 can be aligned, G_3 and G_4 can be aligned, then the new resultant graphs can be $G_{1,2}$ and $G_{3,4}$ respectively. Then align $G_{1,2}$ and $G_{3,4}$ together and continue this process until there is no more sequences to be aligned. If this can be done, then it could take less run time than the current procedure. So this is another sight to look on in the future.

Algorithm 1 Mapper in Cloud POA

```
seqno ← 0
fasta ← ReadFasta(args.infile)
graphBase ← poaGraph()
for (label, sequence) in fasta do
    alignment ← seqGraphAlignment()
    graphBase.incorporateSeqAlignment()
end for
alignments = graphBase.generateAllAlignments()
END =0
```

Algorithm 2 GENERATE_CONSENSI(S, G, w)

```
while (sequences left to be bundled) do
    P ← HEAVIEST_BUNDLE(G,w)
    C_b ← CREATE_SEQUENCE_ON_PATH(P,G)
    I ← ADD_SEQUENCES_TO_BUNDLE(S,B,G,C_b)
    if I is NULL then
        break
    end if
    RESCALE_WEIGHTS(I,w)
    b++
end while
RETURN list of C_b
END =0
```

VIII. Conclusion

For applying and development of novel graph algorithms to search for biologically fascinating features in sequence data, the PO-MSA representation is itself helpful. MSAs can be used to find and analyze phylogenetic relationships through

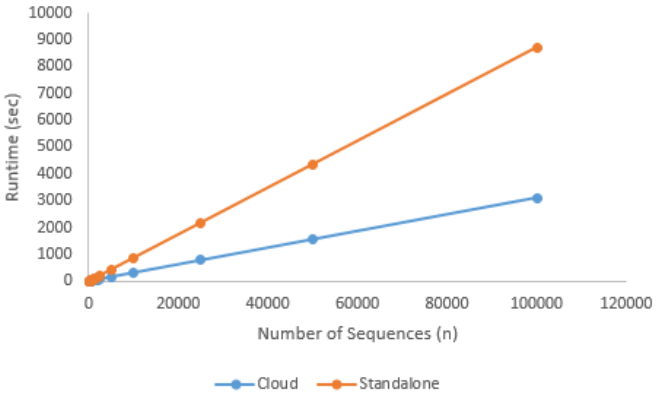

Fig. 4. Standalone VS Cloud POA

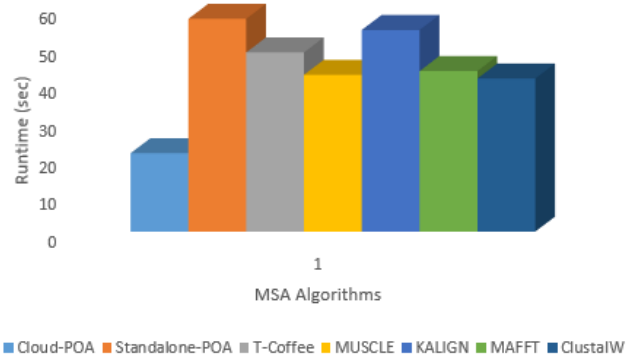

Fig. 5. Comparison of Major MSA Algorithms with Cloud-POA

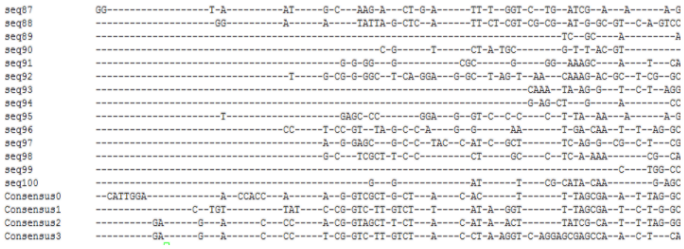

Fig. 6. Final Alignment and Consensus Sequences

homology between sequences. Point mutations including deletion or insertion events (called InDels) can be detected too. After completing all the experiments, a significant reduction in terms of run time was observed which eventually provided a better result than all other sequence alignment algorithms. Moreover, it also upholds the expected accuracy. Despite the fact that there are some works that can be done in future to improve the current implementation, it is already showing very promising results. Hopefully, the current work along with all future improvements can assist in extracting better knowledge concerning sequence alignment which can play a vital role in terms of identity, similarity, homology, evolutionary relationship and many other biological aspects.

REFERENCES

[1] Hogeweg P, Hesper B . "The alignment of sets of sequences and the construction of phyletic trees: an integrated method". J Mol Evol. 20 (2): 17586.

[2] Mount DM. (2004). Bioinformatics: Sequence and Genome Analysis 2nd ed. Cold Spring Harbor Laboratory Press: Cold Spring Harbor, NY.

[3] Higgins DG, Sharp PM (1988). "CLUSTAL: a package for performing multiple sequence alignment on a microcomputer". Gene. 73 (1): 237244.

[4] Thompson JD, Higgins DG, Gibson TJ (1994). "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice". Nucleic Acids Res. 22 (22): 46734680. PMC 308517

[5] "EMBL-EBI-ClustalW2-Multiple Sequence Alignment"

[6] Hirosawa M, Totoki Y, Hoshida M, Ishikawa M (1995). "Comprehensive study on iterative algorithms of multiple sequence alignment". Comput Appl Biosci. 11 (1): 1318.

[7] Gotoh O (1996). "Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments". J Mol Biol. 264 (4): 82338.

[8] B. Morgenstern, DIALIGN: multiple DNA and protein sequence alignment at BiBiServ, Nucleic Acids Research, vol. 32, supplement 2, pp.W33W36, 2004.

[9] R. C. Edgar, MUSCLE: multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Research, vol. 32, no. 5, pp. 17921797, 2004.

[10] Christopher Lee, Catherine Grasso and Mark F. Sharlow, "Multiple Sequence Alignment using Partial Order Graphs," Bioinformatics, vol. 18, pp. 452–464, March 2002.

[11] Christopher Lee , Generating consensus sequences from partial order multiple sequence alignment graphs, (2003)

[12] Ramu Chenna Hideaki Sugawara Tadashi Koike Rodrigo Lopez Toby J. Gibson Desmond G. Higgins Julie D. Thompson, Multiple sequence alignment with the Clustal series of programs, Nucleic Acids Research, Volume 31, Issue 13, 1 July 2003, Pages 34973500.

[13] Jurate Daugelaite, 1 Aisling O Driscoll, 2 and Roy D. Sleator1, Review Article: An Overview of Multiple Sequence Alignments and Cloud Computing in Bioinformatics, Received 24 May 2013; Accepted 23 June 2013.

[14] Yu-Jung Chang, Chien-Chih Chen, Chuen-Liang Chen and Jan-Ming Ho, A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework, Published online 2012 Dec 7. doi: 10.1186/1471-2164-13-S7-S28.

[15] Michael C. Schatz, CloudBurst: highly sensitive read mapping with MapReduce, Bioinformatics. 2009 Jun 1; 25(11):1363-9. doi: 10.1093/bioinformatics/btp236. Epub 2009 Apr 8.

[16] Tung Nguyen, Weisong Shi and Douglas Ruden, CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping, BMC Research Notes20114:171

[17] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Google.inc.

[18] Nelesen S1, Liu K, Zhao D, Linder CR, Warnow T., The effect of the guide tree on multiple sequence alignments and subsequent phylogenetic analyses., Pac Symp Biocomput. 2008:25-36.

[19] http://www.fruitfly.org/sequence/human-datasets.html, 21 August, 2017